JAVASCRIPT & AJAX

Create Bookmarklets - The Right Way

by Siddharth *17 Feb 2011*

Bookmarkets can be defined as mini applications masquerading as tiny snippets of JavaScript awesomeness that reside in your browser and provide additional functionalities to a web page.

Today, we're going to look into creating bookmarklets from scratch and on the way, some best practices to follow.

> *They're not for the nerdy, in fact, they're pretty user-centric.*

We're always looking at ways to enhance our browsing experience, be it widespread or little known. Bookmarklets, I used to think, belonged to the latter - the bastion of super nerds. Much to my chagrin, I discovered that I was completely wrong about the topic. They're not for the nerdy, in fact, they're pretty user-centric, implement a lot of pretty neat functionalities and, just as people predicted, it's become a core part of the way I interact with the browser -- and the internet.

Today, I'd like to walk you through developing bookmarklets implementing some nifty bookmarks. Yes, *bookmarks.* We'll be building more than one, albeit quite tiny ones. Intrigued? Let's get started!

So What Exactly are Bookmarklets?

To quote my article intro:

> *Bookmarkets can be defined as mini applications masquerading as tiny snippets of JavaScript awesomeness that reside in your browser and provide additional functionalities, with just a click.*

The word itself is a blend, *portmanteau* for the pedantic, of the words bookmark and applet. Also called favelets, these tiny snippets of JavaScript let you conjure up additional functionality for any page you're visiting.

Because they consist of just JavaScript, they're pretty portable -- they're supported by all browsers on all platforms, even on mobiles and tablets. Installing them is as simple as dragging them to their toolbar!

Yeah, But What's the Point?

The point is that bookmarklets let you do a lot of things that are otherwise too developer-centric to achieve. Every functionality you get with a bookmarklet can be duplicated using a browser's console and a little time. Bookmarklets merely simplify the process -- packaging up the code that implements some functionality inside a neat little button.

Bookmarklets can be broadly categorized into a number of categories:

- Ones that pass on data. Bookmarklets that submit a page to a service fall under this category. Bookmarklets dealing with social media, dictionary lookups, searches all fall under this category. We'll be building one that submits to Reddit.
- Ones that obtain information and/or modify the current page. We'll be building a bookmarklet that sets a page's

background color.
- Ones that works behind the scenes. A bookmarklet that erases any cookies of the current site is a prime example and one that we'll be building.

#1 - Getting Started

The first point that you'll need to remember is to prefix all JavaScript code with the `javascript` URI. Browsers have implemented this specific prefix so that the content followed by it can be treated, and parsed, as proper JavaScript code.

For example, clicking this link - see its code below - will alert a text.

```
<a href="javascript: alert('Linky text');">Linky</a>
```

#2 - Use an Anonymous Function as a Wrapper

Remember that your code will run against the currently loaded page -- a page that probably will have JavaScript of its own and all the resulting collisions that it implies. The last thing that you need is to have your bookmarklet break the current page.

Wrapping up your code neatly inside an anonymous function will make sure that there are zero naming collisions. Additionally, JavaScript noobs will be flummoxed and will think you're a god, if you're into that sort of thing.

```
javascript:(function(){// your code here })();
```

This is pertinent when you're using JavaScript elsewhere as well. Always, *always* insulate your code.

#3 - Externalize, if Needed

A bookmarklet needn't necessarily be tiny, you're free to go as big as you want. In those cases, in the interest of distribution and keeping your code upto date without manual user intervention, it's best to create a wrapper that pulls in code as needed.

```
javascript: (function () {
    var jsCode = document.createElement('script');
    jsCode.setAttribute('src', 'http://path/to/external/file.js');
  document.body.appendChild(jsCode);
 }());
```

That's pretty much what the above code does -- it creates a script element, sets the source to a file hosted elsewhere and finally appends it to the document. This way, whenever a part of your code changes you can deploy your changes to a single source and have it instantly propagate to every user.

> *Note: You aren't limited to doing this for JavaScript alone. If your bookmarklet makes use of a front end, you're free to pull in the HTML and CSS externally as well, making your bookmarklet truly self updating.*

#4 - Exercise Caution when Adding a Library

Again, if you're building a behemoth of a bookmarklet, you may have need for one of those JavaScript libraries. But using them in your page isn't as simple as merely including it -- you'll need to make sure that the libraries isn't already available. Libraries

like jQuery and MooTools dominate the market and you'd do well to make sure it's not already loaded.

Another issue is that the page may have loaded another library which may lead to conflicts over the control of the $ symbol. Version collisions may also come into play at some point so keep that in mind as well.

Here is a quick shell for the code I generally use. Remember, in your production code, you'll need to account for the issues I've mentioned above.

```
if (!($ = window.jQuery)) { // typeof jQuery=='undefined' works too
    script = document.createElement( 'script' );
   script.src = 'http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js';
    script.onload=releasetheKraken;
    document.body.appendChild(script);
}
else {
    releasetheKraken();
}

function releasetheKraken() {
    // The Kraken has been released, master!
    // Yes, I'm being childish. Place your code here
}
```

The code should be fairly self explanatory. Let's run through it quickly anyway.

- First up, we determine whether jQuery is loaded by checking whether the jQuery object exists in the namespace.
- If not, we quickly prepare it for inclusion. We follow best practices and load it up from a CDN. Finally make sure to point to the main function containing the code to be executed.
- If it's already included, merely run the main wrapper function.

> *If you're looking to avoid all this hassle, I highly recommend Ben Alman's excellent bookmarklet generator. It takes care of namespace and versioning conflicts in a very clean manner. Good stuff!*

#5 - Don't Mess up the Parent Page, Unless you have to

> *Bookmarklets are useless if they, unintentionally, break a page.*

I can't stress this point enough. Bookmarklets are useless if they, unintentionally, break a page. And worrying about JavaScript isn't the only thing you have to deal with.

If you have a front end, the HTML and CSS comes into play as well. Do not assign your containers and classes generic names -- if you name it 'container', you'll forever have my undying hate.

An easy way is to prefix [or suffix, I don't mind] everything with a bookmarklet specific string. And when writing your CSS, be very, very specific. Making use of cascading is nice but use it with ultra precision. Having styles leak over to the main page is irregular and doesn't evoke confidence.

#6 - Test, Test, Test

If you're building a smaller bookmarklet, one where including a third party library seems pointless, you'll run into an ever present demon -- cross-browser compatibility.

Might seem pretty obvious but this is something a lot of people forget, a lot of times. Make sure you test across as many

browsers on as many platforms as possible.

Another easy trap to fall into is a bookmarklet that is meant to work on all sites that only works selectively. Pages can have different hierarchies and make use of different methodologies. Some sites may embrace HTML5 and use the related containers while other may play it safe and use generic divs. While mining for information, make sure to account for every school of development.

And Off We Go

Now that we've looked at some points to keep in mind during development, we're going to build the three bookmarklets I talked about previously -- one belonging to each category.

- A bookmarket that submits to Reddit
- A bookmarket that clears cookies of the current website
- A bookmarklet that changes a page's background color

If you're already familiar with basic JavaScript, the following code should be a cinch-- they're only a few lines of code! The bookmarklets below have been with me for quite some time now and I can't seem to find out where I originally saw them. If you do, leave me a quick note in the comments and I'll add credits accordingly.

Bookmarklet #1 - Submit to Reddit

We'll start off with a super easy one -- submitting to an external service, Reddit in our case.

```
javascript:location.href='http://www.reddit.com/submit?url='
                        +encodeURIComponent(location.href)
                        +'&title='+encodeURIComponent(document.title)
```

This is some pretty simple stuff, taken right from the site. No anonymous wrapper or namespacing since the functionality is pretty limited.

When the bookmarklet is invoked, the following takes place, in logical order:

- The page's title and URL are captured using `location.href` and `document.title`
- To make sure no illegal characters sneak in, we encode it using the `encodeURIComponent` method
- Finally, we concatenate everything to get our URL and change the page's location to it

This pattern works pretty much for most searching and submitting purposes. The only variable here would be the URL you submit to -- each application takes a different approach. Under a minute of searching should see you through.

For example, a validating bookmarklet would look like so:

```
javascript:location.href='http://validator.w3.org/check?uri='+encodeURIComponent(location.href)
```

Note that we merely discarded the page's title and changed the URL structure to point to the W3 validation service.

Bookmarklet #2 - Change a Page's Background

```
javascript:void(document.bgColor=prompt('Color? Hex or name will do. May I suggest pink?','pink'))
```

This is just as simple as it looks. To keep things super simple, I've opted to not introduce variables and validation. We'll just look at the core code.

- **First up**, we create a prompt that asks the user for a color to set the background to. Since we're not checking the input, I'm relying on you not being naughty.
- **Finally**, we set the background to the chosen color through `document.bgColor`. That's it!

Yeah, I know `bgColor` is deprecated and yeah, I know I should feel dirty but this will do quite adequately since a quick test on modern browsers came out positive.

You can play around with these quite a bit. This is similar to how you'd generally apply CSS styling to elements through JavaScript. We're just applying it to arbitrary pages. Do make note that different pages have different structures so if the page has a full wrapping container, the effect may be meaningless. In these cases, you may have to hunt for the main container and then apply the styling to it.

Advertisement

Bookmarklet #3 - Clearing cookies

```
javascript:function(){
    var c=document.cookie.split(";");
    for(var i=0;i<c.length;i++){
        var e=c[i].indexOf("=");
        var n=e>-1?c[i].substr(0,e):c[i];
        document.cookie=n+"=;expires=Thu, 01 Jan 1970 00:00:00 GMT";
    }
}()
```

Not really as complicated as it looks -- cookies merely have a very strict syntax so we'll need to parse it step by step. Let's go through the code:

- Split document.cookie based on semicolors. `c` now becomes an array containing all the cookies set for the current domain.
- Loop through the array.
- Check whether the cookie contains the equal to symbol i.e. whether a cookie value has been set and returned
- If yes, we'll capture all information just upto it. If not, let the cookie be as it is
- Finally, add a string value which empties the cookie's value along with defining an expiry date from the past, effectively erasing it

Remember, bookmarklets still operate under the page so you can't access cookies from other domains. You're limited to the page you invoke the bookmarklet on.

That's a Wrap

And we're done. Remember, we've merely scratched the surface of bookmarklets. A lot of them are complete applications within themselves, with a proper front end to boot. Those take a lot of time and planning to build, just like any non-trivial application. This was merely to get your foot in the door of bookmarklet development. Hopefully you've found this tutorial interesting and this has been useful to you.

Questions? Nice things to say? Criticisms? Hit the comments section and leave me a comment. Thank you so much for reading!

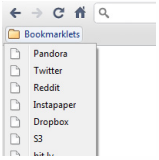
Advertisement

Difficulty:
Beginner

Length:
Short

Tagged with:

JavaScript & AJAX | Web Development | Javascript



About Siddharth

N/A

View Full Profile

+ Expand Bio

Advertisement

Related Posts

Using Polymer to Create Web Components

31 Mar 2014 • Polymer will help you to easily create Web Components for your web apps, using a similar syntax t...

Creating a WordPress Theme From Static HTML: Releasing Your Theme

18 Mar 2014 • If you've been following this series you now have a working WordPress theme. Your theme has a num...

Connect 4 With Socket.io

4 Mar 2014 • Today we'll see how we can use Node.js and Socket.io to create a multiplayer Connect 4 style game.

JavaScript Tools of the Trade: JSBin

4 Feb 2014 • We've all been there. There are times when you simply want to throw some JavaScript code up and s...

JavaScript Objects

1 Jan 2014 • In JavaScript, objects are king: Almost everything is an object or acts like an object. Understan...

Working With IndexedDB

13 Sep 2013 • One of the more interesting developments in web standards lately is the Indexed Database (Indexed...

Ghostery a bloqué les commentaires publiés par Disqus.

Advertisement

tuts+

Teaching skills to millions worldwide.

About

Blog

FAQ

Support

Write For Us

Advertise

Privacy Policy

General T&Cs

Courses T&Cs

© 2013 Envato Pty Ltd.

envato